A real-time calibration method based on time-to-digital converter for accelerator timing system*

Qihao Duan,[1,2] Liang Ge,[1,†] Yanhao Jia,[1,2] Jieyu Zhu,[1,2] and Wei Zhang[1,2,‡]

[1]*Institute of modern physics, Chinese Academy of Sciences, Lanzhou 730000, China*
[2]*University of Chinese Academy of Sciences, Beijing 100049, China*

The high-intensity heavy-ion accelerator facility (HIAF) is a scientific research facility complex composed of multiple cascade accelerators of different types, which pose a scheduling problem for devices distributed over a certain range of 2 km, involving over a hundred devices. The White Rabbit (WR), a technology-enhancing Gigabit Ethernet, has shown the capability of scheduling distributed timing devices but still faces the challenge of obtaining real-time synchronization calibration parameters with high precision. This study presents a calibration system based on a time-to-digital converter implemented on an ARM-based System-on-Chip (SoC). The system consists of four multi-sample delay lines, a bubble-proof encoder, an edge controller for managing data from different channels, and a highly effective calibration module that benefits from the SoC architecture. The performance was evaluated with an average RMS precision of 5.51 ps by measuring the time intervals from 0 to 24000 ps with 120000 data for every test. The design presented in this study refines the calibration precision of the HIAF timing system. This eliminates the errors caused by manual calibration without efficiency loss and provides data support for fault diagnosis. It can also be easily tailored or ported to other devices for specific applications and provides more space for developing timing systems for particle accelerators, such as white rabbits on HIAF.

Keywords: HIAF, White Rabbit, Calibration system, Time-to-digital converter (TDC)

I. INTRODUCTION

Time, one of the seven fundamental physical quantities in physics, has been extensively studied and applied in various fields, such as large-scale physics experiments, lunar exploration projects, defense industries, 5G communications, and navigation systems. To explore fundamental particles in the microscopic world, scientists have increasingly demanded requirements for the performance of particle accelerators. Particle accelerators are multisystem [1–4], highly complex, and strongly coupled systems characterized by a wide variety of devices, dispersed placements, and large spatial spans. Compared with other complex systems, particle accelerator systems have extremely stringent timing requirements, with some reaching the femtosecond level [5, 6].

Leading scientific and technological powers worldwide attach great importance to nuclear physics research based on particle accelerators, evident in the construction of large-scale scientific facilities, the development of powerful experimental detection devices, and the internationalization of research projects and teams. The research team at the Institute of Modern Physics, Chinese Academy of Sciences is currently constructing a national major science and technology infrastructure called the "the High Intensity Heavy Ion Accelerator Facility " (HIAF) [7–9] as shown in Fig. 1.

HIAF is a heavy-ion scientific research facility with leading international capabilities and wide-ranging applications [10–12]. Its primary scientific goals include understanding effective interactions within atomic nuclei, investigating the origins of elements ranging from iron to uranium in the universe, studying the properties of high-energy-density matter, and addressing key technologies related to particle irradiation. The HIAF consists of several components, including a Superconducting Electron Cyclotron Resonance Ion Source (SECR), superconducting linear accelerator (iLinac), Booster Ring (BRing), radioactive secondary beam- separation device (HFRS), high-precision ring spectrometer (SRing), and experimental terminals [13, 14]. The iLinac injector injects various ions from protons to uranium for BRing. BRing, a room-temperature synchrotron accelerator, is the core component of HIAF and lays the foundation for obtaining high-intensity, high-energy, and high-quality heavy-ion beams. After BRing accelerates the beam, it is either extracted directly or slowly to the experimental terminal or injected into the high-precision ring spectrometer SRing through the radioactive secondary beam separation device for related experiments.

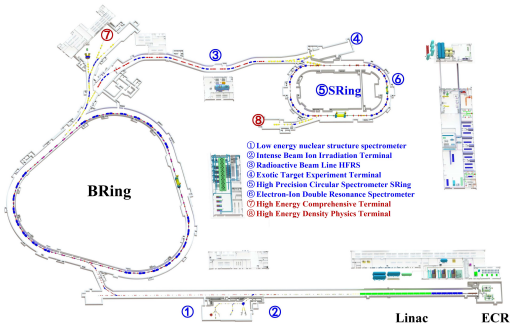


Fig. 1. Layout of the accelerator complex of HIAF.

To achieve a higher energy and beam intensity, a common approach in particle accelerators is to cascade multiple accelerators of different types, where the accelerator that boosts the beam energy in the previous stage serves as the injector

* This work is supported by High Intensity heavy-ion Accelerator Facility (HIAF) approved by the National Development and Reform Commission of China (2017-000052-73-01-002107)
† Corresponding author, gliang016@impcas.ac.cn
‡ Corresponding author, impzw@impcas.ac.cn

for the subsequent stage. In the case of HIAF, after the ion source generates the beam, it is accelerated through a superconducting linear accelerator (iLinac) before being injected into BRing. The cascading of multiple accelerator stages allows for an increase in the beam energy while enabling the parallel operation of the accelerators.

The beam undergoes acceleration through a series of interconnected accelerators and is eventually directed to the experimental terminal for the relevant experiments. This poses a scheduling problem for devices distributed over a certain range, where the goal is to optimize the scheduling to achieve lossless injection, accumulation, acceleration, and beam extraction. In this case, the scheduling variables form an n-dimensional time vector.

The timing system prototype of HIAF is based on the White Rabbit (WR) protocol and achieves timing scheduling with a precision of better than 2 ns. However, it also faces challenges in calibrating and monitoring the timing devices distributed across a range of 2 km involving over a hundred devices. Synchronization calibration of the timing system is a complex process. Offline calibration can be achieved, and the synchronization status can be queried once the devices are online. However, deviations in synchronization cannot be fed back in real-time, thereby preventing real-time synchronization calibration. [15] proposed a distributed time-to-digital converter in a white rabbit network to capture the arrival times of shower particles and produce unified timestamps of all particles. This gave us the opportunity to construct a high-resolution, real-time calibration system based on a time-to-digital converter. Many works have achieved high-precision time-to-digital converters and applied these techniques in various applications[16–21], especially for physical researches. However, information regarding the implementation details of time-to-digital converters is scarce. The objective of this study is to address these issues.

The main contributions of this study are as follows.

1) We proposed a real-time calibration system based on the White Rabbit protocol for the HIAF timing system.

2) We proposed a calibration architecture for time-to-digital converter in an ARM-based System-on-Chip (SoC) with high development efficiency.

3) We proposed a series of detailed modules to implement a time-to-digital converter for lower technical barriers in this area.

4) We implemented and tested our real-time synchronization calibration system based on the time-to-digital converter in a ZYNQ board (a series of SoCs produced by Xilinx).

II. WRFM

The core components of the HIAF timing system include the Clock Master Node (WRCM), Data Master Node (WRDM), Synchronization Network (WRNT), Terminal Nodes (WRN), White Rabbit Switches (WRS), and various online services. The structure of the system is illustrated in Fig. 2.

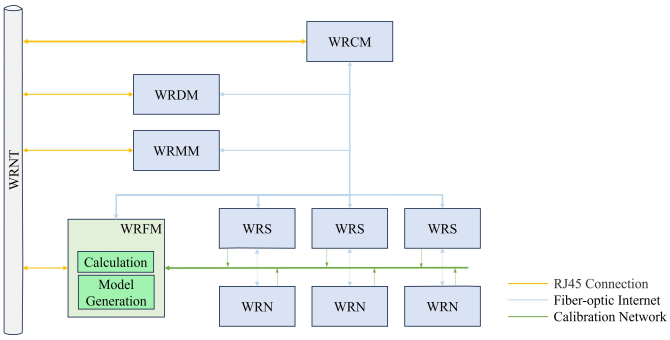


Fig. 2. The timing system on HIAF.

In this timing system, all the clocks of these components are synchronized with WRCM. After receiving the clock and current time signals from WRCM, each node produces a Pulse Per Second (PPS) output. This calibration system aims to ensure that the PPS signals generated by different devices are synchronized with WRCM, representing the time synchronization of these devices.

At the beginning of the calibration, the framework of the monitor (WRFM) gathers basic information on the round-trip delays between WRCM and WRS or WRS and WRN, transmission times, and receipt times of all nodes, such as $delay_{MM}$, Δ_{TXM}, Δ_{RXM}, Δ_{TXS} and Δ_{RXS}. The objective of calibration is to measure and update the stored transmission and reception times within the network to match real-world measurements, ensuring that the timing system generates PPS signals simultaneously after adjustment [22].

The WRFM responsible for realizing system-wide synchronization monitoring, synchronization parameter calculation, and deviation model generation were built using time-to-digital converter (TDC) technology. The deviation statistics module was implemented using dedicated hardware, whereas the online calculation and model generation modules were implemented on servers.

Owing to the shorter development cycle and stronger support for some communication protocols, the deviation statistics module is implemented in ZYNQ, which calculates the time deviation between the output signals (PPS signals) of the timing system components and the local output signals from WRFM as a time-to-digital converter. The structure is illustrated in Figs. 3.

Combining the set threshold and multiple sets of time deviation statistics, the module triggers the online calculation module according to predefined rules. The online calculation module computes the synchronization parameters and updates them accordingly. The model generation module generates device-level or system-level models based on statistical time deviations, thereby providing a foundation for system optimization.

The calibration system follows Eqs. 1 and 2 [22].

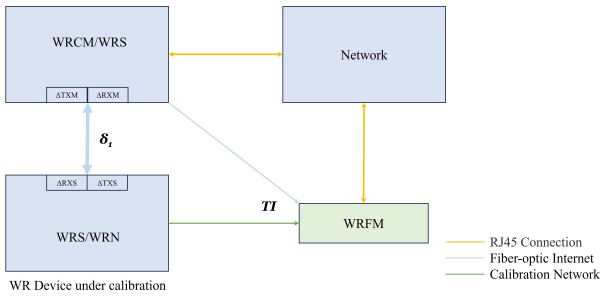


Fig. 3. The localized structure of the calibration system.

$$\frac{1}{2}\Delta_S = \frac{1}{2}(delay_{MM} - \Delta_{TXM} - \Delta_{RXM} - \epsilon_S - \delta_1) \quad (1)$$

and

$$\begin{cases} \Delta_{TXS} &= \frac{1}{2}\Delta_S - TI \\ \Delta_{RXS} &= \frac{1}{2}\Delta_S + TI \end{cases} \quad (2)$$

where $delay_{MM}$ represents the round-trip delay between WRCM, WRS, and WRN. Additionally, Δ_{TXS} denotes the transmission delay of the nodes, Δ_{RXS} is the reception delay of the nodes, δ_1 is the latency of the fiber connecting nodes, ϵ_S is the compensation value of the nodes when Δ_{TXS} is zero, and TI is the time interval obtained from the deviation statistics module (TDC) illustrated at Sec.II A.

The key focus of WRFM is the deviation statistics module (TDC) because its accuracy determines the overall system accuracy.

A. Architecture of TDC

The intuitive idea behind implementing a TDC based on FPGA (field-programmable gate arrays) is to employ a counter that runs at the system clock rate. However, the granularity of the system counter could not satisfy the requirements of white rabbits. Therefore, it is necessary to obtain subclock-period resolution. The proposed algorithm is illustrated in Fig. 4.

It comprised a set of start-and-stop channels. The hit signals existing as one start hit and one-stop hit latched by the system clock are interpreted as subclock fine timestamps from the two corresponding channels, whereas the coarse counter clocked by the system clock outputs the coarse timestamp. The starting and stopping timestamps are defined as follows:

$$timestamp_{start} = m * T_{WR} + \tau_{start} \quad (3)$$

and

$$timestamp_{stop} = n * T_{WR} - \tau_{stop} \quad (4)$$

where T_{WR} is the period of the coarse counting clock of the White Rabbit system, m and n are the coarse timestamps

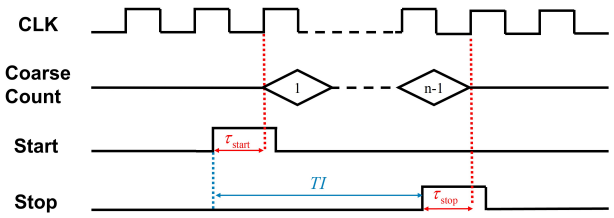


Fig. 4. Basic TDC Algorithm.

from the coarse counter, whereas τ_{start} and τ_{stop} are timestamps corresponding to the respective fine counter channels. Hence, the TI can be calculated as:

$$\begin{aligned} TI &= (timestamp_{\text{stop}} - timestamp_{\text{start}}) \\ &= (\tau_{\text{start}} - \tau_{\text{stop}}) + (m - n) \times T \end{aligned} \quad (5)$$

An organic combination of the two types of timestamps comprised the final measurement result.

Fig. 5 shows the system architecture of the proposed TDC implemented in ZYNQ. The system consists of programmable logic(PL) and a processing system (PS), which benefits from the real-time advantages of FPGA and ARM's flexibility. The PL part is responsible for TDC's mainstay, including the tapped delay lines (TDLs), a D flip-flop bank, a thermometer-to-binary encoder, an edge controller, and data First In, First Out (FIFO). The PS is responsible for calibration logic and communication with a personal computer (PC) through a universal asynchronous receiver/transmitter interface. An advanced extensible interface (AXI) is the data path between PL and PS components. Every part of our system architecture is interpreted below to elaborate further on our system architecture.

1. *Delay Line*

A typical method for increasing the granularity of TDC is to interpolate more basic cells into one primitive system clock period. Thus, the delay line is one of the core elements in the TDC design, which defines the system's resolution and linearity. This depends on the type of the basic delay cell used as the interpolation unit. The most common delay elements in FPGA platforms are CARRY4 cell primitives (fast carry logic with look-ahead) because they have dedicated routing with the smallest internal propagation delay [23].

The TDLs in this study employed cascade-carrying elements.

The hit signal propagates through the delay chain by connecting to the CYINIT port of the first delay cell and linking the last bit of the CO to the next cell's CI port as the Fig. 6).

According to [24]

, we can determine that the inner path time of a complete CARRY4 logic is significantly shorter than that of the coarse clock, which is approximately 60 ps. A delay line can be constructed by placing these cells sequentially. The coupled

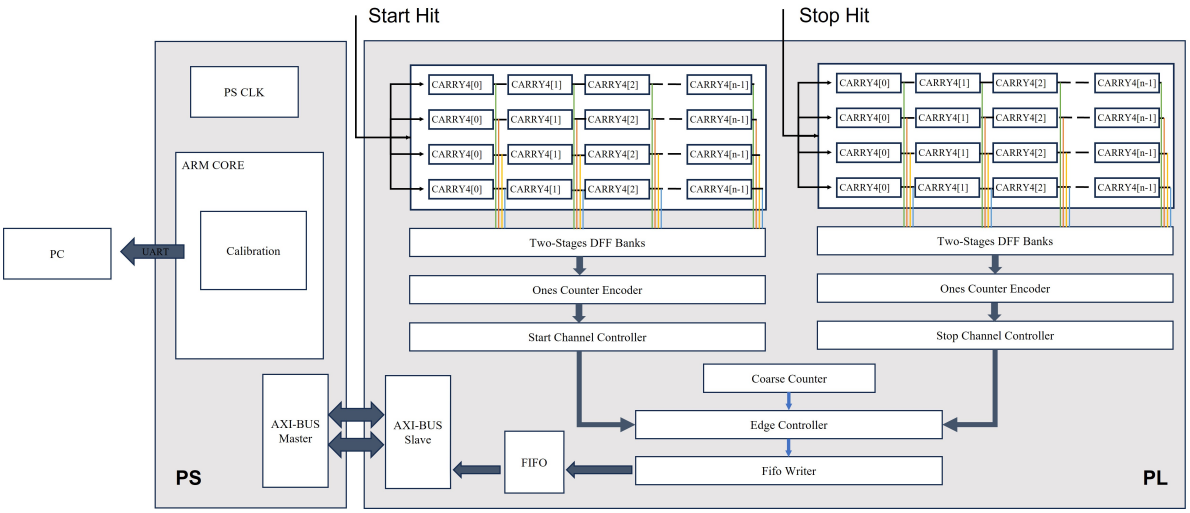


Fig. 5. TDC System.

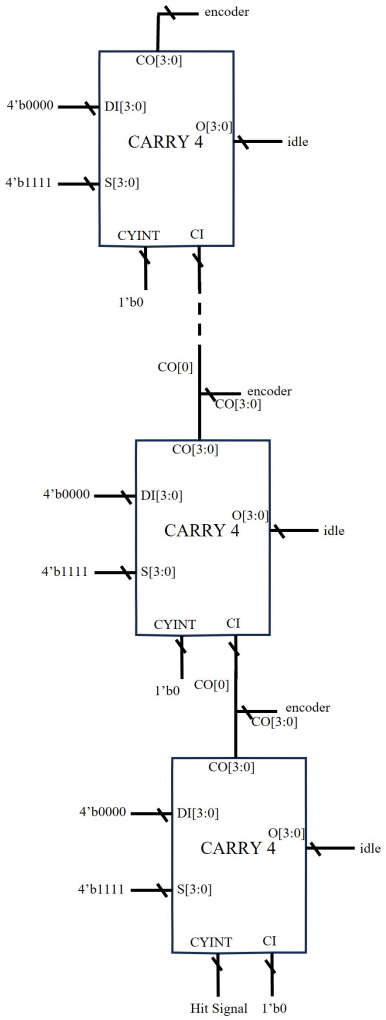


Fig. 6. Delay Line Structure.

two-stage D flip-flops tap out the status of the hit signal in the delay line to reduce the possibility of a metastable state.

There are tricks when placing delay chains on a physical board at the implementation stage that could be the key to increasing the stability of the delay chains.

First, the closer the entrance of a delay chain to the physical input/output port, the narrower the bin width of the first cell. Second, the delay chain should be placed within a certain clock region to reduce the harm caused by time skew when crossing the clock region, which would interfere with the accuracy of the sampling phase. Therefore, the system clock is crucial for balancing stability and accuracy. This design considered a 500 Mhz clock frequency divided from the White Rabbit reference clock and a TDL with 200 delay cells.

Third, in the case of a multiline TDC, the gap space between the lines belonging to one channel is unnecessary. Comparative experiments revealed that the introduced gap caused transfer-time delays.

The input signal was fed simultaneously into four parallel chains to improve the time resolution beyond the intrinsic cell delay. This involves sampling a specific timespan four times, increasing the granularity by a factor of four because it characterizes a physical quantity with more quantities.

We collected all taps from the four delay chains and processed them as they originated from a common delay chain.

2. *Ones-Counter Encoder*

The output of the TDL, which is a thermometer code representing the time interval, must be converted into a binary number. One of the classical ways to achieve this is to instinctively detect the transition of the 0-1 position in delay lines [25, 26]. However, the time delay used to register the tap should be considered, which includes not only delays of cells but also time skews of the inter-and outer time zones. The deviation between the sample and real values is intro-

duced, which is the most severe problem for TDC implementation: the bubble problem. Ideally, the output of the TDL should be a clean thermometer code such as 1111110000. Because of uneven propagation delays among delay lines and the tapped register's time difference, the bubble problem appears and disturbs the thermometer code; for example, instead of 1111110000, the thermometer sampled out is 1111010100. When generating a correct binary code, the bubble problem induces hassles in classical 0-1 transition detection encoders because the primitive TDLs cannot tap out an ideal thermometer code. This is more difficult for FPGAs with 28 nm and more advanced process technology [27]. Collecting several delay-line taps at once certainly loses the order consistent with the real delay, owing to the reasonable variance of the transition time at the same position from different lines will add more factors leading to bubbles. Consequently, the bubble problem with several delay lines is more severe than a single line.

Therefore, designing a bubbleproof encoder is essential. Lui and Wang proposed a bin-realignment technique to remove bubbles using a tap-swapping method before sampling the primitive TDL code from the encoder. However, the bin-realignment method is complicated and requires at least two cycles of FPGA synthesis for a complete tap-order calibration procedure with a PC at initialization. This is also time-consuming during the runtime [28, 29]. Inspired by the solution implemented in [30], a one-counter encoder is adopted in this study as a robust bubble-proof encoder.

As mentioned previously , the bubble problem is caused by the disorder of taps when they are transported from the delay lines to the encoder. Therefore, the "1" and "0" are sufficient to accurately represent the time it takes for the hit signal to propagate in this system, no matter the taps' sequence. This also applies when the taps from different delay lines are used. The tap values for each delay line represent the corresponding propagation times. When the hit signal was collectively fed into these delay lines, the tap values in each delay line effectively underwent multiple samplings of the same signal, thereby increasing the precision of the measurement results. This enhancement led to better granularity and resolution.

The ones-counter is an intuitive way to add all the tap values together for counting "1"s in delay lines. However, it is essential to consider the actual computational performance. Through experiments, we found that it is unfeasible to directly add all tap values together at once because the adder is composed of cascaded look-up tables (LUTs) in the FPGA (PL), and the time consumption is a combination of computations within a certain stage of cascaded LUTs and transportation between stages. Therefore, adopting a step-by-step calculation method for the counter was necessary. The computational module is shown in Fig. 7. We implemented a computational module to implement a step-by-step calculation method in one counter. We grouped the primitive tap values from all four delay lines into sets of six elements that were added together using LUT-6, a type of primitive cell on the Xilinx Development Board. The output of the six-element adder is then transformed into a 3-bit binary form by setting specific parameters in the LUT-6s. Next, we sum the 3-bit binary values from every pair of groups to obtain a 4-bit binary value. This process was repeated for 10 stages, resulting in the outcome of a 10-bit binary value sent to the edge controller for synthesis.

3. Channel Controller

After encoding the time interval into binary form, the resulting data indicate the current propagation time within the delay lines. However, when a hit signal occurs, a binary value is generated and changes during propagation. We proposed the module shown in Fig. 8, which uses a state machine structure to address this issue. The module has two states: state-ready, which is an idle mode waiting for a hit signal and assigning the binary value from the encoder to a new variable, and state-keep, which is a mode that keeps the data received at the beginning of the hit signal. If we attempt to maintain the data after both the hit signal and the change to state-keep, a pattern delay could occur, causing the data to become outdated. Therefore, assigning and maintaining state-ready data in another state is better. The locked data are then transferred to the edge-controller module for further computation.

4. Edge Controller

The edge controller module is a core component of the TDC and is responsible for managing the time interval data from both fine counters, including the start and stop channels and the coarse counter.

The coarse counter counts the time interval using a digital counter running on the system clock and the control logic.

The measurement span depends on the coarse bit width; a wider bit width results in a broader range.

We use a flag to control the coarse counter, which starts and stops counting in a pipeline-like manner.

When the start signal is high, and the stop signal is low, the coarse counter switches to counter mode and increments by one on every system clock. After the stop signal switches to active, regardless of the state of the start signal, the coarse counter switches to the keep mode and remains in this mode until the stop signal switches back to inactive, thus completing the overall calculation.

The edge controller logic requires more state machines than its coarse counterparts. It has four states: state-ready, state-readout-fine, state-output, and state-wait-for-ending, as shown in Fig. 9. In the state-ready mode, the module is idle before receiving a start-hit signal. The state-readout fine captures the real-time output of the two channels and sets two flags to declare. One path was reserved for sending data at the state readout, and it was fine. If there is an unanticipated delay during transportation, the module switches to the wait-for-end state until completion. This processing logic is simple but useful when dealing with sequential missions. Subsequently, the data generated from the start and stop channels can be transported to the next stage for a time interval transformation.

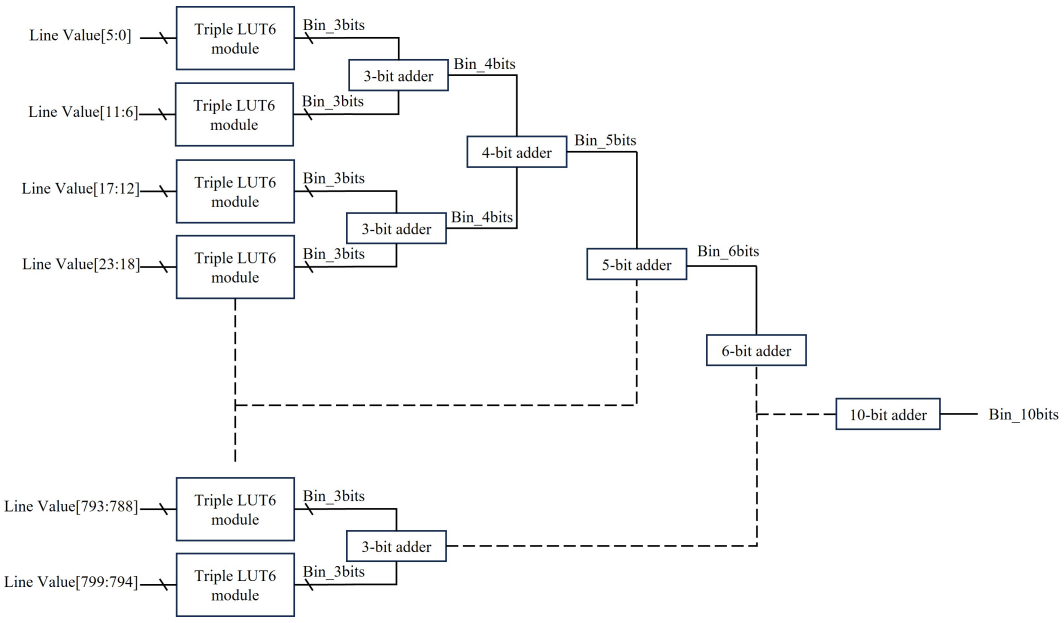


Fig. 7. Ones-counter Encoder.

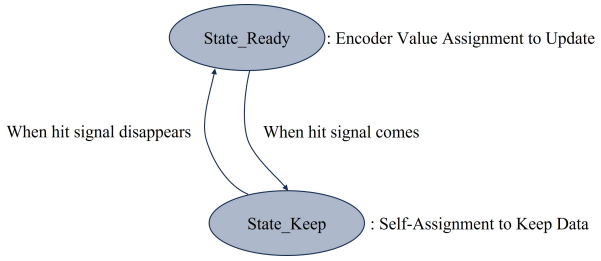


Fig. 8. Channel Controller.

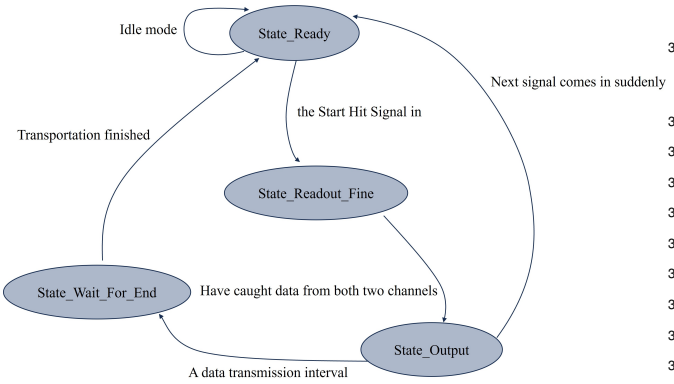


Fig. 9. Edge Controller.

5. *Calibration and Output*

The output from the edge controller module is stored as 32-bit data containing a 10-bit coarse count, 10-bit start channel count, and 10-bit stop channel count. To transform these primitive binary count values into actual time intervals, we must consider that the delay times of each delay cell are different. A significant error will occur if we multiply the count value by a fixed bin width. The key to this process is determining the width of each bin by adding the bin sizes through which the hit signal has propagated through a process called calibration. After the calibration, the time interval between the start and stop hit signals was calculated using the algorithm shown in Fig. 4.

Conversion from bin numbers to picosecond are as follows [31, 32]:

$$T_i = \sum_{n=1}^{i-1} W_n + \frac{W_i}{2} \tag{6}$$

where T_i represents the measured time interval of the hit signal propagated through the i bins. W_n is the corresponding width of the n th bin.

TDC calibration mechanisms are often required for modern FPGAs, and the bin-by-bin calibration method has been widely used to enhance the linearity of TDC [28]. Typically, TDCs based on FPGA require several steps to implement this function. First, they constructed a connection interface, such as a Universal Asynchronous Receiver/Transmitter (UART) and Peripheral Component Interconnect Express (PCIE), etc. on the FPGA and sent primitive count values to the PC for analysis of the bin width. Then, the construction of the time mapping is implemented through storage media, such as block random access memory (BRAM), which requires a time period to initiate. Finally, when a new group of fine count values arrives, it serves as an index for determining the corresponding value stored in the BRAM in advance. The result is then output through the communication interface to a PC [17].

However, the process is complicated, time-consuming, and requires a long time to initiate the system.

Therefore, we propose a new calibration method for ZYNQ, as shown in Fig. 5 to improve the development efficiency. The calibration mechanism on the PS section.

After obtaining the actual bin widths through the required code density test method [24, 33], which was introduced in IV A, the calibration maps were transformed into an array form of the C language using Python and inserted into the codes of the PS part. This significantly decreases the time consumption for initialization, which is required to initialize BRAM before the system runs. The 32-bit primitive count data, combined with the coarse count, count from the start channel, and count from the stop channel, are stored in a FIFO on the FPGA for later transmission. The AXI transports these data from PL to the PS, which are read from FIFO without omission, even at different system clocks. The coupled primitive data were disassembled into a coarse count, starting count, and stopping count at the ARM.

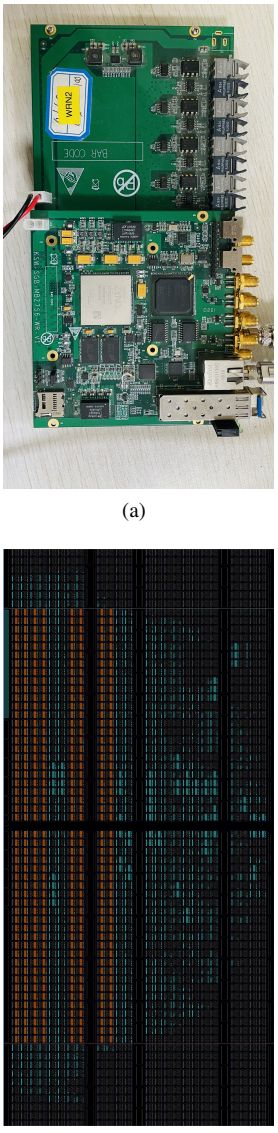
The calibration process involved inserting these count values into predefined calibration map arrays. Finally, the readable measured time interval is sent to the PC through the UART.

III. IMPLEMENT DETAILS

A. Time Sequence Control

Time-sequence control is a critical aspect of developing an FPGA program. We used many primitive design cells, such as the D Flip-Flop with Clock Enable and Asynchronous Clear (FDCE), to precisely control the signal flow tempo. The stages of FDCE should be the same as the number of paths that one logical calculation requires to maintain logical health.

It is also essential to determine the time required for a logical calculation before setting the system clock to ensure that the logical operation requirements are met. Methods also exist to address time errors when a high system clock is required for certain systems. For example, dividing a complicated logical calculation module into several pieces running simultaneously is a commonly used approach, such as in this study.



(a) (b)

Fig. 10. (a) the self-developed board and (b) the implemented block diagram.

IV. PERFORMANCE EVALUATION

We implemented a real-time synchronization calibration system on a ZYNQ-7000 self-developed board and tested its performance through time interval measurement ability experiments. The ZYNQ-7000 self-developed board and the implemented block diagram are shown in Fig. **??**.

To evaluate the performance of the core TDC, we placed two channels (start and stop channels in Fig. 5) to minimize the offset. In addition, we utilized an arbitrary waveform generator (model AFG3252) from Tektronix as an external signal source. The same square-wave signal produced by the generator was simultaneously fed into two channels via two subminiature-version-a connector (SMA) connections to reduce measurement errors and jitter from cables connecting the signal resource and evaluation board. The frequency was selected to ensure the completeness of the hit signal. The time interval between hit signals was adjusted by modifying the phase difference between the two output channels. When hit signals were detected, the TDC recorded both channels' coarse and fine timestamps, which were read by a PC via the Universal Asynchronous Receiver/Transmitter interface of the ARM part on the board.

A. Bin Width and Resolution

The bin width is a quantifiable indicator of the physical delay chain and represents the actual time interval of one delay cell.

The TDC bin widths can be measured using a code density test, in which the output of the wave generator is controlled such that its frequency is not correlated with the system clock. The hit signal can be treated as a random signal for the two channels because the arrival time is not fixed according to the asynchronous rhythm of the TDC's sample time. Because of the equal probability of the hit signal arrival time during one clock period, the corresponding frequency of the hit signal detected in one TDC bin reflects the TDC bin width.

According to the number of hits collected in the x-th bin, the corresponding TDC bin width can be calculated as follows:

$$W_x = \frac{H_x \times T_{sys}}{H_{total}} \tag{7}$$

where W_x is the bin width of the xth bin. H_{total} is the number of random hits. And H_x is the number of hits that proliferate within a certain bin. T_{sys} denotes the clock period of the system.

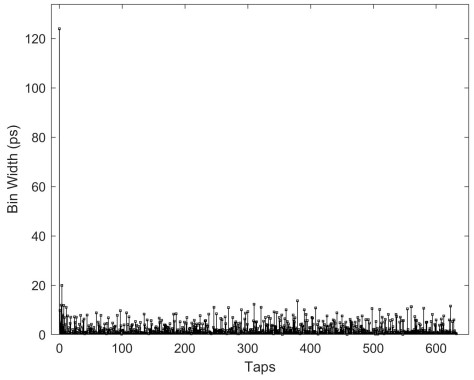
We set the hit signal emission frequency to 20.11111 Mhz, which is approximately unrelated to the system clock at 500 Mhz (with a clock period of 2 ns) and at least 120000 hits as one data set to calculate the bin width for calculation robustness.

During code density tests, we discovered that the distance between the entrances of the hit signal could have a subtle impact on the widths of the first and last bins. If the distance is too small, the width of the first few bins will be zero, which can lead to dissatisfaction with cell placement within one clock region unless the system clock frequency is increased.

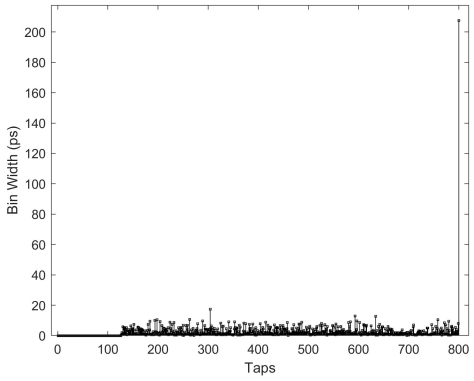
However, this could affect the system stability and make the last bin too large. In contrast, if the distance is too large, the TDC's first delay bin width will be too large to represent the actual time interval accurately.

The experimental results for the situations too close to and too far away are shown in Fig. 11. The first and second bins widths were 123.89 ps in Fig. 11(a). The final bin width was 207.64 ps in Fig. 11(b), which was unsatisfactory.

Usually, we cannot determine the physical positions of the input IOs (input/output) on an already designed board, but pursuing a sweet point for placement is still necessary. After multiple adjustments, we found a suitable location for the delay lines. Fig. 12 shows the final code-density test results. The effective bins of the start channel begin at bin 72 with a width of 0.0765 ps and end at bin 798 with a width of 0.2142 ps. The effective bins of the stop channel begin at bin 41 with a width of 6.7936 ps and end at bin 798 with a width of 0.0765 ps. By interpolating these bins into one system clock period (2 ns), a higher resolution can be achieved, with an average of 2.75 ps and 2.71 ps for two channels, respectively.



(a)



(b)

Fig. 11. Measured bin widths as (a) remotely placed and (b) close placed.

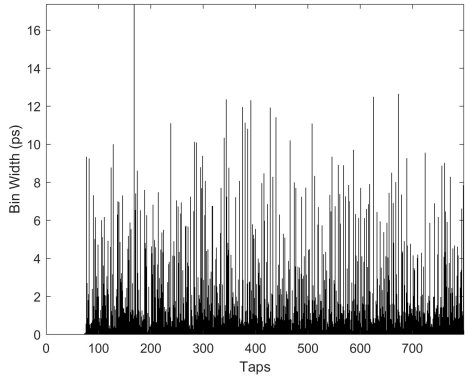
The differential nonlinearity (DNL) and integral nonlinearity (INL) can be deduced from the measured bin widths in Fig. 12, and both are used to describe nonlinearity. DNL is defined as every bin deviation from the average bin width, whereas INL is defined as the collected deviation of the current bin by summing the deviation values before it. The calculation method for DNL and INL can be expanded as follows:

$$DNL_x = \frac{W_x - W_{ave}}{LBS} \tag{8}$$

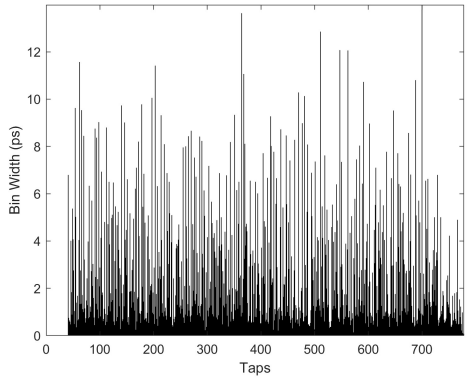
and

$$INL_x = \sum_{j=0}^{x} DNL_j \tag{9}$$

where DNL_x is the DNL of the x-th bin and W_x is the x-th bin width. Correspondingly, w_{ave} is the average channel bin width. The equation of INL is easily understood. The measured DNL and INL of the start channel are -0.99 to 5.30 LSB (the least significant bit) and -6.99 to 17.86 LSB as shown in Fig. 13(a) and Fig. 13(b). And that of the stop channel are -0.98 to 4.15 LSB and -2.10 to 17.86 LSB, respectively, as shown in Fig. 13(c) and Fig. 13(d).

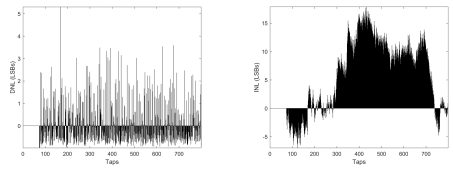


(a)

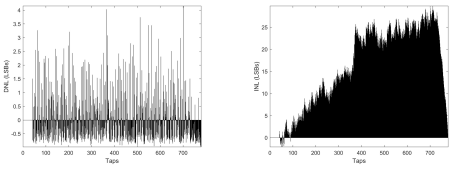


(b)

Fig. 12. Bin width of (a) the start channel and (b) the stop channel.



(a) (b)



(c) (d)

Fig. 13. Measured DNL and INL of the start channel (a, b) and the stop channel (c, d).

The INL indicates the error when treating the average bin width as the real bin width. Hence, bin-by-bin calibration is essential to solve this problem, as mentioned in Section. II A 5. The final results after calibration without INL were used as the measurement results.

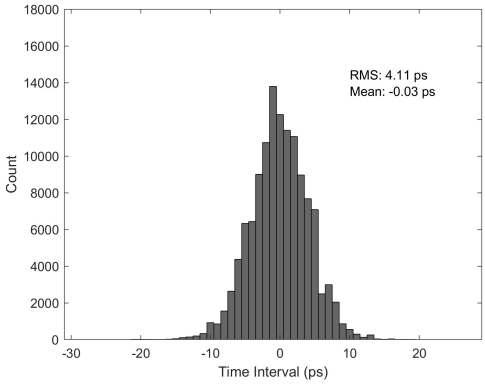


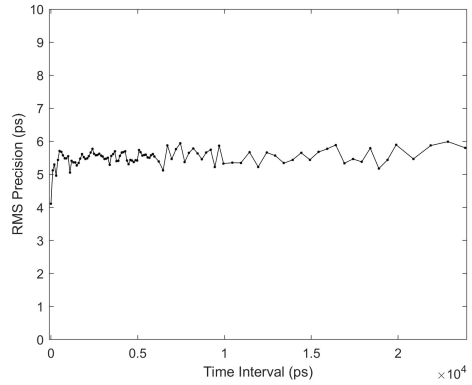
Fig. 14. Measured RMS at 0 ps.

B. Time interval tests

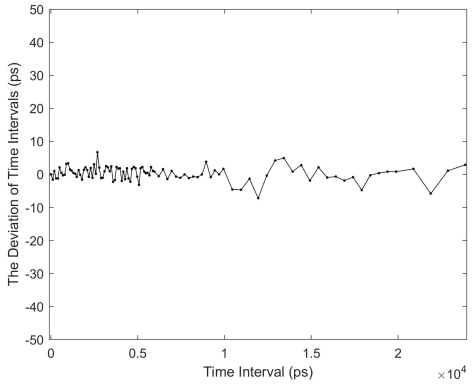
The root mean square (RMS) represents the measurement uncertainty introduced by jitter and quantization errors [34]. It is evaluated using multiple time-interval measurements for a certain time interval, generated by adjusting the output phase of one channel on the wave generator. The calculation method is followed as Equ. 6. Similar to the bin-width tests, we considered 120000 test data points as one data set. The RMS histogram with a typical normal distribution tested at 0 ps is shown in Fig. 14.

We conducted a series of time-interval tests ranging from 0 ps to 24000 ps. To balance the stride and range, we used a test step of 100 ps in the range of 0-6000 ps, 250 ps in the range of 6000-10000 ps, 500 ps in the range of 10000-20000 ps, and 1000 ps in the range of 20000-24000 ps. The results are shown in Fig. 15.

The best RMS performance appears when the time interval is 0 ps, achieving 4.11 ps RMS precision, and deteriorates slightly after that. Upon checking the primitive counter values, we found that the closer the time interval is to the 0 ps, the less likely the coarse counter is to engage in the final time calculation. Only a few measurements were obtained with the coarse counter in the repetitive measurement of the time interval near 0 ps, which represents the lower jitter of the coarse counter will be introduced to the result. This occurs only when the starting hit signal arrives at the end of one coarse counter period in the start channel, and the stopping hit signal emerges at the beginning of the coarse counter period in the stop channel. However, even for a micro signal, it is still difficult to ensure the arrival time; hence, the coarse counter will always be considered. The measured time interval value of about 342.78 ps at 0 ps can be considered as the offset time of this system, resulting from the length deviation of the two input signal cables and the pathway length required for the two-channel signals to cross. This is because these internal factors introduce only a delay at 0 ps. A later time interval result was obtained by subtracting the measured value from the offset value. As shown in Fig. 15, the RMS precision ranges from about 5.0 ps to 5.9 ps with an average of 5.5 ps, and the deviation from the corresponding time is in a range of less

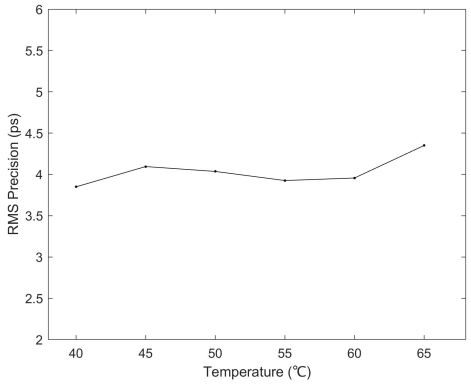


(a)

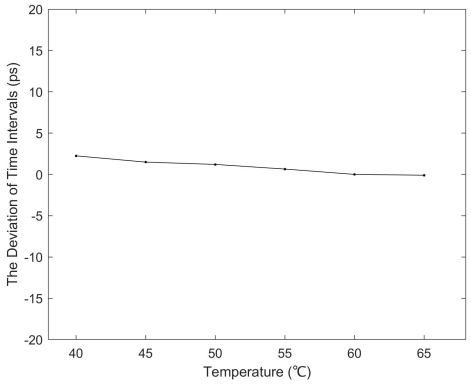


(b)

Fig. 15. (a) The RMS precision and (b) the deviation values from the corresponding time in a range from 0 ps to 24000 ps.



(a)



(b)

Fig. 16. (a) the Measured RMS and (b) the deviation values at 0 ps as the temperature changing from $40°C$ to $65°C$.

than 10 ps, which is acceptable as a requirement of the white rabbit system.

C. Temperature

Generally, there is a close connection between temperature and FPGA performance. Therefore, it is essential to test this system at different temperatures. The general working temperature ranged from $40°C$ to $65°C$; therefore, we used a hair dryer to heat the board and maintain the temperature with the help of an electric fan.

Because the transmission speed of the hit signal differed at different temperatures, we generated a calibration table at $60°C$, which already covered the longest pathway record in one delay line. The test results are presented in Fig. 16. Performance changed with temperature. The best RMS precision appears at $40°C$ at approximately 3.85 ps because this temperature is the most suitable for this board. The second-best RMS precision appears at $55°C$ about 3.93 ps, which is better than that at $60°C$ because the calibration is set at $60°C$ near $55°C$, and lower temperature will make FPGA more linear. After heating the system from $40°C$, the performance began to deteriorate until it reached a turning point in the middle of the $45°C$ to $50°C$. The deviations in the time intervals were less than the average RMS precision of the system. This demonstrates the system's robustness when the calibration table is set to a suitable state.

D. Logic Resources Consumption

Table. 1 summarizes the resource utilization in the two-channel system. The data extracted from the implementation report by Vivado (2018) demonstrated low resource consumption and good potential for multichannel applications.

Table 1. Logic Resources Utilization.

Resource	Utilization	Available	Utilization(%)
LUT	3194	171900	1.86
LUTRAM	66	70400	0.09
FF	6826	343800	1.99
BRAM	4	500	0.8
IO	4	250	1.6
PLL	1	8	12.5

V. CONCLUSION

In particle accelerators, a common approach to achieve higher energy and beam intensity is to cascade multiple accelerators of different types. In the case of HIAF, the beam generated by the ion source is accelerated through a superconducting linear accelerator (iLinac) before being injected into BRing. The beam undergoes acceleration through a series of interconnected accelerators. It is eventually directed to the experimental terminal for relevant experiments, which poses scheduling problems for distributed devices over a certain range and a real-time calibration challenge for the timing system.

This paper describes a novel architecture of a real-time calibration module used for the White Rabbit timing system, which can achieve high-resolution online calibration for different subunits. We introduce a multiline time-to-digital converter based on an ARM-based System-on-Chip (SoC) as the core calibration component, with a novel edge controller and a highly effective calibration module that benefits from the SoC architecture. The hardware implementation of this system is described in detail. The experimental results indicate that the proposed calibration system is suitable for 5.51 ps precision calibration missions, even in extreme environments.

The design presented in this study refines the calibration precision of the HIAF timing system. This eliminates the errors caused by manual calibration without efficiency loss and provides data support for fault diagnosis. It can also be easily tailored or ported to other devices for specific applications and provides more space for the development of timing systems for particle accelerators, such as white rabbits on HIAF.

[1] T. Liu, H. S. Song, Y. H. Yu et al., Towards real-time digital pulse process algorithms for CsI(Tl) detector array at External Target Facility in HIRFL-CSR. Nucl. Sci. Tech. **34**, 131 (2023). doi: 10.1007/s41365-023-01272-6.

[2] H. M. Xie, K. W. Gu, Y. Wei et al., A non-invasive Ionization Profile Monitor for transverse beam cooling and orbit oscillation study in HIRFL-CSR. Nucl. Sci. Tech. **31**, 40 (2020). doi: 10.1007/s41365-020-0743-7

[3] Z. S. Li, X. J. Yin, H. Du et al., The IH-RFQ for HIRFL-CSR injector. Nucl. Sci. Tech. **29**, 89 (2018). doi: 10.1007/s41365-018-0416-y

[4] F. F. Duan, Y. Y. Yang, B. T. Hu et al., Silicon Detector Array for Radioactive Beam Experiments at HIRFL-RIBLL. Nucl. Sci. Tech. **29**, 165 (2018). doi: 10.1007/s41365-018-0499-5

[5] N. Akasaka, A. Akiyama, S. Araki et al., KEKB accelerator control system. Nucl. Instrum. Meth. A. **499**, 138–166 (2003). doi: 10.1016/S0168-9002(02)01786-2

[6] Carlos Megías, Víctor Vázquez, Eduardo Ros et al., Ethernet-based timing system for accelerator facilities: The IFMIF-DONES case. Comput Netw. **233**, 109897 (2023). doi: 10.1016/j.comnet.2023.109897

[7] Y. Yang, L. T. Sun, Y. H. Zhai et al., Heavy ion accelerator facility front end design and commissioning. Phys. Rev. Accel. Beams. **22**, 110101 (2019). doi: 10.1103/PhysRevAccelBeams.22.110101

[8] M. T. Tang, L. J. Mao, H. J. Lu et al., Design of an efficient collector for the HIAF electron cooling system. Nucl. Sci. Tech. **32**, 116 (2021). doi: 10.1007/s41365-021-00949-0

[9] P. Yu, B. Zhang, F. Wang et al., Fabrication and cold test of prototype of spatially periodic radio frequency quadrupole focusing linac. Nucl. Sci. Tech. **32**, 1–9 (2021). doi: 10.1007/s41365-020-00835-1

[10] P. Liu, J. H. Chen, Y. G. Ma et al., Production of light nuclei and hypernuclei at High Intensity Accelerator Facility energy region. Nucl. Sci. Tech. **28**, 55 (2017). doi: 10.1007/s41365-017-0207-x

[11] J. H. Liu, Z. Ge, Q. Wang et al., Electrostatic-lenses position-sensitive TOF MCP detector for beam diagnostics and new scheme for mass measurements at HIAF. Nucl. Sci. Tech. **30**, 152 (2019). doi: 10.1007/s41365-019-0676-1

[12] G. F. Qu, W. P. Chai, J. W. Xia et al., Two-plane painting injection scheme for BRing of HIAF. Nucl. Sci. Tech. **28**, 114 (2017). doi: 10.1007/s41365-017-0260-5

[13] J. C. Yang, J. W. Xia, G. Q. Xiao et al., High Intensity heavy ion Accelerator Facility (HIAF) in China. Nucl. Instrum. Meth. B. **317**, 263–265 (2013). doi: 10.1016/j.nimb.2013.08.046

[14] Y. Liu, D. Q. Gao, H. J. ZHANG et al. Design and implementation of HIAF-Kicker power supply communication system. NUCLEAR TECHNIQUES **44** 31–38 (2021). doi: 10.11889/j.0253-3219.2021.hjs.44.070402.

[15] W. B. Pan, G. H, Gong, Q. Du et al., High resolution distributed time-to-digital converter (TDC) in a White Rabbit network. Nucl. Instrum. Meth. A **738**, 13–19 (2014). doi: 10.1016/j.nima.2013.11.104

[16] M. Parsakordasiabi, I. Vornicu, Á. Rodríguez-Vázquez et al., A Low-Resources TDC for Multi-Channel Direct ToF Readout Based on a 28-nm FPGA. Sensors. **21**, 308 (2021). doi: 10.3390/s21010308

[17] J. Y. Wu, Z. H. Shi et al., The 10-ps wave union TDC: Improving FPGA TDC resolution beyond its cell delay. 2008 IEEE Nuclear Sci. Conf. R. (IEEE, Dresden, 2008), pp. 3440–3446. doi: 10.1109/NSSMIC.2008.4775079

[18] Y. Xiao, H. G. Liang, L. Y. Jin et al., Design of a high-accuracy time-to-digital converter based on dual-edge signals. J. Instrum. **18**, P07023 (2023). doi: 10.1088/1748-0221/18/07/P07023

[19] Z. W. Xue, H. Liang, J. X. Li et al., A 128-channel time-to-digital converter based on the 24-phase shift-clock sampling method for RPC. J. Instrum. **18**, T02007 (2023). doi: 10.1088/1748-0221/18/02/T02007

[20] L. N. Cojocariu, D. Foulds-Holt, F. Keizer et al., A multi-channel TDC-in-FPGA with 150 ps bins for time-resolved readout of Cherenkov photons. Nucl. Instrum. Meth. A. **1055**, 168483 (2023). doi: 10.1016/j.nima.2023.168483

[21] D. Yang, Z. Cao, X. J. Hao et al., Readout electronics of a prototype time-of-flight ion composition analyzer for space plasma. Nucl. Sci. Tech. **29**, 60 (2018). 10.1007/s41365-018-0390-4

[22] G. Daniluk, White Rabbit Calibration Procedure. Accessed: Mar. 2018. [Online]. Available at: https://www.ohwr.org/documents/213.

[23] R. Machado, J. Cabral, F. S. Alves, Recent Developments and Challenges in FPGA-Based Time-to-Digital Converters. IEEE T. Instrum. Meas. **68**, 4205–4221 (2019). doi:

10.1109/TIM.2019.2938436

[24] J. Torres, A. Aguilar, R. García-Olcina et al., Time-to-Digital Converter Based on FPGA With Multiple Channel Capability. IEEE T. Nucl. Sci. **61**, 107–114 (2014). doi: 10.1109/TNS.2013.2283196

[25] M. Choi and A. Abidi., A 6-b 1.3-Gsample/s A/D converter in 0.35-/spl mu/m CMOS. IEEE J. Solid-St. Circ. **36**, 1847–1858 (2001). doi: 10.1109/4.972135

[26] V. E. Garuts, Y. S. Yu, E. O. Traa et al., A dual 4-bit 2-Gs/s full Nyquist analog-to-digital converter using a 70-ps silicon bipolar technology with borosenic-poly process and coupling-base implant. IEEE J. Solid-St. Circ. **24**, 216–222 (1989). doi: 10.1109/4.18579

[27] Y. G. Wang, X. Y. Zhou, Z. Q. Song et al., A 3.0-ps rms Precision 277-MSamples/s Throughput Time-to-Digital Converter Using Multi-Edge Encoding Scheme in a Kintex-7 FPGA. IEEE T. Nucl. Sci. **66**, 2275–2281 (2019). doi: 10.1109/TNS.2019.2938571

[28] C. Liu, Y. G. Wang, A 128-Channel, 710 M Samples/Second, and Less Than 10 ps RMS Resolution Time-to-Digital Converter Implemented in a Kintex-7 FPGA. IEEE T. Nucl. Sci. **62**, 773–783 (2015). doi: 10.1109/TNS.2015.2421319

[29] Y. G. Wang, C. Liu, A 3.9 ps Time-Interval RMS Precision Time-to-Digital Converter Using a Dual-Sampling Method in an UltraScale FPGA. IEEE T. Nucl. Sci. **63**, 2617–2621 (2016). doi: 10.1109/TNS.2016.2596305

[30] Y. G. Wang, J. Kuang, C. Liu et al., A 3.9-ps RMS Precision Time-to-Digital Converter Using Ones-Counter Encoding Scheme in a Kintex-7 FPGA. IEEE T. Nucl. Sci. **64**, 2713–2718 (2017). doi: 10.1109/TNS.2017.2746626

[31] E. Bayer, P. Zipf, and M. Traxler, A High-Resolution (< 10 ps RMS) 48-Channel Time-to-Digital Converter (TDC) Implemented in a Field Programmable Gate Array (FPGA). IEEE T. Nucl. Sci. **58**, 1547–1552 (2011). doi: 10.1109/TNS.2011.2141684

[32] E. Bayer, P. Zipf, M. Traxler, A multichannel high-resolution (< 5 ps RMS between two channels) Time-to-Digital Converter (TDC) implemented in a field programmable gate array (FPGA). 2011 IEEE Nuclear Sci. Conf. R. (IEEE, Valencia, 2011), pp. 876–879. doi: 10.1109/NSSMIC.2011.6154560

[33] J. Y. Wu, Several Key Issues on Implementing Delay Line Based TDCs Using FPGAs. IEEE T. Nucl. Sci. **57**, 1543–1548 (2010). doi: 10.1109/TNS.2010.2045901

[34] J. Kalisz, Review of methods for time interval measurements with picosecond resolution. Metrologia. **41**, 17 (2003). doi: 10.1088/0026-1394/41/1/004